In the Specification

Please amend the paragraph beginning at line 16 of page 4 as follows:

-- It should be understood, that in practical applications the graphs typically include thousands of nodes connected in any number of different ways, and containing many ~~loops~~ cycles. Analyzing such graphs to determine optimal configurations is difficult.--

Please amend the paragraph beginning at line 22 of page 6 as follows:

-- The average probability of failure for BP decoding over many blocks is now considered. A real number p_{ia}, which represents the probability that the message m_{ia} is an erasure, is associated with each message m_{ia}. Similarly, a real number q_{ai}, which represents the probability that the message m_{ai} is an erasure, is associated with each message m_{ai}. In the density evolution method, probabilities p_{ia} and q_{ai} are determined in a way that is exact, as long as the Tanner graph representing the error-correcting code has no ~~loops~~ cycles.--

Please amend the paragraph beginning at line 4 of page 7 as follows:

-- The equation for p_{ia} is

$$p_{ia} = x \prod_{b \in N(i) \backslash a} q_{bi} , \qquad (3)$$

where $b \in N(i)\backslash a$ represents all check nodes directly connected to a neighboring variable node i, except for check node a. This equation can be derived from the fact that for a message m_{ia} to be an erasure, the variable node i must be erased in transmission, and all incoming messages from other check nodes are erasures as

well. Of course, if the incoming messages are correlated, then this equation is not correct. However, in a Tanner graph with no ~~loops~~ cycles, each incoming message is independent of all other messages.--

Please amend the paragraph beginning at line 19 of page 7 as follows:

-- The density evolution equations (3) and (4) can be solved by iteration. A good initialization is $p_{ia} = x$ for all messages from variable nodes to check nodes and $q_{ai} = 0$ for all messages from check nodes to variable nodes, as long as the iteration begins with the m_{ai} messages. The BEC density evolution equations ultimately converge. This can be guaranteed for codes defined in graphs without ~~loops~~ cycles. It is possible to determine b_i, which is the probability of a failure to decode at variable node i, from the formula

$$b_i = x \prod_{a \in N(i)} q_{ai} . \qquad (5)\text{--}$$

Please amend the paragraph beginning at line 5 of page 8 as follows:

-- As stated above, the density evolution equations (3, 4, and 5) are exact when the code has a Tanner graph representation without ~~loops~~ cycles.--

Please amend the paragraph beginning at line 12 of page 11 as follows:

-- The intuitive reason that these equations are valid, in the infinite block-length limit, is that as $N \to \infty$, the size of typical ~~loops~~ cycles in the Tanner graph of a regular Gallager code go to infinity, so all incoming messages to a node are independent, and a regular Gallager code behaves as a code defined on a graph

without ~~loops~~ cycles. Solving equations (34 and 35) for specific values of d_v and d_c yields a solution that is $p = q = b = 0$, below a critical erasure limit of x_c. This means that decoding is perfect. Above x_c, b has a non-zero solution, which correspond to decoding failures. The value x_c is easy to determine numerically. For example, if $d_v = 3$ and $d_c = 5$, then $x_c \approx 0.51757$.--

Please amend the paragraph beginning at line 5 of page 12 as follows:

-- Unfortunately, the density evolution method is erroneous for codes with finite block-lengths. One might think that it is possible to solve equations (3 and 4) for any finite code, and hope that ignoring the presence of ~~loops~~ cycles is not too important a mistake. However, this does not work out, as can be seen by considering regular Gallager codes. Equations (3, 4, and 5) for a finite block-length regular Gallager code have exactly the same solutions as one would find in the infinite-block-length limit, so one would not predict any finite-size effects. However, it is known that the real performance of finite-block-length regular Gallager codes is considerably worse than that predicted by such a naive method.--

Please amend the paragraph beginning at line 3 of page 14 as follows:

-- Figure 1 is a prior art bipartite graph representing an error-correcting code including a ~~loop~~ cycle;

Figure 2 is a prior art bipartite graph representing a simple prior art error-correcting code;

Figures 3a-e are bipartite graphs renormalized according to the invention;

Figure 4 is a bipartite graph to be renormalized according to the invention;

Figure 5 is a bipartite graph with ~~loops~~ cycles to be renormalized;

Figure 6 shows an expansion of a bipartite graph to be renormalized;

Figure 7 is a bipartite graph with a ~~loop~~ cycle representing a generalized parity check matrix to be renormalized;

Figure 8 is a graph comparing methods of evaluating error-correcting codes;

Figure 9 is a flow diagram of a renormalization group method according to the invention; and

Figure 10 is a flow diagram of the method according to the invention for a graph with ~~loops~~ cycles.--

Please amend the paragraph beginning at line 21 of page 17 as follows:

-- The RG Transformation for a Decorated Tanner Graphs with no ~~Loops~~ Cycles

First, we consider loop-free Tanner graphs, and determine the RG transformations that are sufficient to give exact failure rates for such error-correcting codes. Then, we extend the RG transformations in order to obtain good approximate results for Tanner graphs with ~~loops~~ cycles.--

Please amend the paragraph beginning at line 25 of page 18 as follows:

-- The renormalization of step 940 is described in greater detail below for ~~loopy graphs~~ graphs with cycles.--

Please amend the paragraph beginning at line 8 of page 20 as follows:

-- This example makes it clear why the RG method is exact for a codeword defined on a graph without ~~loops~~ cycles. The RG transformations essentially reconstruct the density evolution equations of the prior art, and we know that density evolution is exact for such an error-correcting code. The advantage of our RG method is that it gives a much better approximation for error-correcting codes represented by bipartite graphs *with* ~~loops~~ cycles. It is well understood, that good error-correcting codes, even of moderate size, will always have ~~loops~~ cycles, mainly because ~~loops~~ cycles provide redundancy without substantially increasing the size of the code.--

Please amend the paragraph beginning at line 17 of page 20 as follows:

-- **The RG Method for a Graph with ~~Loops~~ Cycles**

For an error-correcting code represented by a graph with ~~loops~~ cycles, we eventually have to renormalize 940 a variable node that is not a "leaf" node. Note that we never have to renormalize a non-leaf check node. To do this, we first collect all the check nodes a, b, etc. connected to the target node i. We discard q_{ai}, q_{bi}, p_{ia}, p_{ib}, etc. For any given check node attached to variable node i, e.g., check

node a, we also collect all the other variable nodes j attached to node a, and renormalize the values of q_{aj}.--

Please amend the paragraph beginning at line 25 of page 22 as follows:

-- We always have an RG transformation for q_{aj} correspond to the logic of the local neighborhood around the variable node i that we are renormalizing. In fact, the RG transformation given in equation (44) is appropriate if the local neighborhood of node i is tree-like, but should be adjusted if there are ~~loops~~ cycles in the local neighborhood.--